



HEALTH LEVEL SEVEN (HL7) UPGRADE

TECHNICAL RELEASE NOTES

Patch DG*5.3*363

May 2002

Department of Veterans Affairs
System Design & Development

Table of Contents

Overview	1
Purpose of This Manual.....	1
Software and Documentation Retrieval.....	1
Routine Changes	2
Direct Access of Message Data into Files #772 and/or #773.....	2
Initialization of HL7 Variables.....	3
Sending the Transmission.....	3
To Transmit Acknowledgement (ACK) Messages	4
To Transmit Generic Non-ACK Messages	4
Storage of Message Before Transmission	5
Message Headers (MSH and BHS)	5
Outgoing Single Message (MSH)	5
Outgoing Batch Message (BHS).....	6
Reading Incoming Messages	6
Protocols	7
Logical Links	8
Routine List with Checksums.....	10

Overview

Health Level Seven (HL7) is a standard protocol that specifies the implementation of interfaces that exchange healthcare information electronically. The **VISTA** HL7 package assists M-based applications to conduct these HL7 transactions by providing the facilities to create, transmit, and receive HL7 messages over a variety of transport layers. The Health Eligibility Center's (HEC) and Veterans Administration Medical Center's (VAMC) current implementation of the **VISTA** HL7 package is Version 1.5, which was the first released version of **VISTA** HL7, and uses VA MailMan as the communications protocol. Version 2.1 of the HL7 industry standard format is currently in use with the **VISTA** HL7 package at the HEC and VAMCs.

In 1993, the Infrastructure Team released **VISTA** HL7 V. 1.6 and recommended that all sites upgrade to this version. **VISTA** HL7 V. 1.6 is compatible with V. 2.1 of the HL7 industry standard format; therefore, an upgrade to V. 2.3.1 of the HL7 industry standard format is not a requirement of this project. Lately, the package upgrade has become more important to the HEC, because the upgrade is necessary to implement the Institution File Redesign and HEC Redesign projects. In order to receive the INSTITUTION File (#4) updates, the HL7 package must contain Patch HL*1.6*57, which is a complete overhaul of the **VISTA** HL7 package and enhances the package's TCP/IP capabilities.

Patch DG*5.3*363 will be bundled into a host file with Patch IVM*2*34. For installation instructions, refer to Patch IVM*2*34.

Purpose of This Manual

The purpose of this manual is to provide technical information about the upgrade to **VISTA** HL7 V. 1.6.

Software and Documentation Retrieval

The software for this patch is not being distributed through the National Patch Module. This patch is being bundled with IVM*2*34 in a host file. Refer to the Software and Documentation Retrieval section of the Patch IVM*2*34 Technical Release Notes and Installation Guide for this information.

Routine Changes

The modifications to the routines listed in this section will update the HL7 interface at the VAMCs to use the new functionality provided by V. 1.6 of the **VISTA** HL7 application.

Direct Access of Message Data into Files #772 and/or #773

Routines affected:
DGENUPL

VISTA HL7 V. 1.6 separates the header of the transmission, and the message of the transmission. The header is placed in File #773, and the message in File #772. Previously, the full transmission was placed in File #772. HLNEXT (a call to HLNEXT^HLCSUTL) was created to gather the full transmission without having to manually traverse the many pointers to all the necessary data. There is a standard procedure (listed in the **VISTA** HL7 Site Managers & Developers Manual) to read all incoming segments of a message, using HLNEXT. The data is stored in the following temporary file:

^TMP(\$J,IVMRTN,SEGCNT,CNT)

{Where: IVMRTN = Main Routine which gathers the incoming message

And: SEGCNT = Incremental Segment Counter CNT = Incremental Continuation Counter Starts at 0}

Only the temporary file should be used by routines to use segment/message data.
Segment/message access should NOT be made directly into File #772 and/or File #773.

Initialization of HL7 Variables

Routines affected:

DGENEGT2

DGENQRY1

DGENUPL1

The new **VISTA** HL7 Messaging V. 1.6 uses the following call:

DO INIT^HLFNC2(HLEID,.HL)

{Where: HLEID = Name or IEN from File #101 of the Event Driver Protocol.

And: HL = HL7 variables returned in the HL array}

Note that commonly used variables such as HLQ, HLFS, and HLECH are also created by this call in order for v1.5 backward compatibility. This is done so that the variables can be set up more specifically for each interface. The Event Driver Protocol will contain information or pointers to information that will describe the who, what and where of the interface.

Sending the Transmission

Routines affected:

DGENEGT2

DGENQRY1

DGENUPL

To Transmit Acknowledgement (ACK) Messages

GENACK^HLMA1(HLEID,HLMTIENS,HLEIDS,HLARYTYP,HLFORMAT,.HLRESLTA,HLMTIENA,.HLP)

{Where: HLEID = Name or IEN of Event Driver Protocol

And: HLMTIENS = IEN of entry in HL7 MESSAGE TEXT File (#772) for subscriber application

HLEIDS = IEN of subscriber event from the PROTOCOL File (#101)

HLARYTYP = Array type. One of the following codes:

- LM = local array containing a single message
- LB = local array containing a batch of messages
- GM = global array containing a single message
- GB = global array containing a batch of messages

HLFORMAT = Format of array, 1 for pre-formatted in HL7 format, otherwise 0

HLRESLTA = The variable that will be returned to the calling application as described above
[PASSED BY REFERENCE]

HLMTIENA = IEN of entry in HL7 MESSAGE TEXT File (#772) where the acknowledgement message will be stored. This parameter is only passed for a batch acknowledgment.

HLP("SECURITY") = A 1 to 40 character string}

To Transmit Generic Non-ACK Messages

GENERATE^HLMA(HLEID,HLARYTYP,HLFORMAT,HLRESULT,HLMTIEN,HLP)

{Where: HLEID = Name or IEN of Event Driver Protocol

And: HLARYTYP = Array type. One of the following codes:

- LM = local array containing a single message
- LB = local array containing a batch of messages
- GM = global array containing a single message
- GB = global array containing a batch of messages

HLFORMAT = Format of array, 1 for pre-formatted in HL7 format, otherwise 0

HLRESULT = The variable that will be returned to the calling application as described above
[PASSED BY REFERENCE]

HLMTIEN = IEN of entry in HL7 MESSAGE TEXT File (#772) where the message being generated is to be stored. This parameter is only passed for a batch type message

HLP("SECURITY") = A 1 to 40 character string

HLP("CONTPTR") = Continuation pointer, a 1 to 180 character string}

Storage of Message Before Transmission

Routines affected:

DGENEGT2
DGENQRY1
DGENUPL
DGENUPL1

In **VISTA HL7 V. 1.6**, the local variable array HLA("HLS",CTR) {Where: CTR = Counter for segments in the message} is used to hold small outgoing transmissions. Large outgoing transmissions are held in the following file:

^TMP("HLS",\$J,CTR)
{Where: \$J = unique job identifier created by M
And: CTR = Counter for segments in the message}

Message Headers (MSH and BHS)

Routines affected:

DGENEGT2
DGENQRY1
DGENUPL1

In **VISTA HL7 V. 1.6**, the HL7 Messaging system routines create the header segment after the application routines have created the message body. In this way, the application routines only create the body of the message.

Outgoing Single Message (MSH)

In **VISTA HL7 V. 1.6**, the application routine will create the outgoing data array or global as described in the Storage of Message Before Transmission section of this document. When the data is passed to the transmission generation routine (GENACK^HLMA1 or GENERATE^HLMA) the MSH header will be created by HL7 Messaging.

Outgoing Batch Message (BHS)

In *VISTA HL7 V. 1.6*, the individual messages (MSH) within the batch are still controlled by the application routine, but the batch header (BHS) is created at the time of transmission by the message transmission routine (GENACK^HLMA1 or GENERATE^HLMA) at the time of transmission. The following are the calls necessary to create a batch transmission:

D INIT^HLFNC2("batch protocol",.HL)
Initializes variables to build the batch message

D CREATE^HLTF(.HLMID,.MTIEN,.HLD1,.HLD1)
Creates a message stub in the HL7 MESSAGE TEXT File (#772) (IEN is returned in MTIEN) and reserves a batch message id (returned in HLMID).

D INIT^HLFNC2("server protocol",.HL)
Initializes variables to build the individual message

D MSH^HLFNC2(.HLARRAY,HLMID_"-n",.HLRES,SECURITY) where n is the batch message counter creates new message header segment for the individual message to append to the batch.

Reading Incoming Messages

Routines affected:
DGENEGT2

When receiving a transmission, HL7 Messaging will place the header in File #773, and the body of the message into File #772. In *VISTA HL7 V. 1.6*, executing HLNEXT (which runs code HLNEXT^HLCSUTL) will cause the entire message to be read from both Files #772 and #773. The following code uses HLNEXT to traverse the transmission and store it.

```
F SEGCNT=1:1 X HLNEXT Q:HLQUIT'>0 D
. S CNT=0
. S ^TMP($J,IVMRTN,SEGCNT,CNT)=HLNODE
. F S CNT=$O(HLNODE(CNT)) Q:'CNT D
. . S ^TMP($J,IVMRTN,SEGCNT,CNT)=HLNODE(CNT)
```

The message is saved into the following temporary global:

^TMP(\$J,IVMRTN,SEGCNT,CNT)

{Where: IVMRTN = Main routine which gathers the incoming message
SEGCNT = Incremental Segment Counter
CNT = Incremental Continuation Counter Starts at 0}

The execution of the HLNEXT variable creates the HLNODE array, which is initialized for each segment read from the incoming message. The HLNODE array consists of the main segment followed by each continuation of that segment. Each series of same segment nodes within the HL7 MESSAGE TEXT File (#772) is separated from the next main segment by an empty node (unlike the V. 1.5 File #772 where there is no continuation of segments and there is no need for empty nodes).

Protocols

Routines affected:

DGENEGT2

DGENQRY1

DGENUPL1

The Protocol Naming Convention:

HEC to VAMC HEC vamc# mtyp-etyt SERVER

VAMC to HEC VAMC vamc# mtyp-etyt SERVER

{where:

vamc# = Station # of the receiving/sending VAMC

mtyp = Message Type (ORU, ORF, QRY, ...)

etyt = Event Type (Z05, Z07, Z11, ...)}

For example,

VAMC 500 ORU-Z07 SERVER (for sending unsolicited Z07 patient demographics from Station 500 to the HEC)

Many of the *VISTA* HL7 V. 1.6 calls need to know the specific event driver protocol that is performing the event.

For example,

INIT^HLFNC2(HLEID,HL)

where HLEID is the IEN of a specific protocol.

Once the full protocol name is formed (look at the Naming Convention section above) the IEN of that specific protocol can be found by using the following code:

To get the IEN of an event driver/SERVER protocol:

```
$O(^ORD(101,"B","VAMC vamc# mtyp-etyt SERVER",0))
```

To get the IEN of a subscriber/CLIENT protocol:

```
$O(^ORD(101,"B","VAMC vamc# mtyp-etyt CLIENT",0))
```

where vamc#, mtyp, and etyp are filled in with actual values

Logical Links

Links (also known as logical links) describe the complete network path to a given system. They are similar in function to VA MailMan's DOMAIN File (#4.2) and Kernel's DEVICE File (#3.5). Link entries hold the details of how to connect to the target system, such as IP address and Port. For every target system that you need to exchange HL7 messages with, a link needs to be set up so that VISTA HL7 knows how to reach the target system.

The naming convention for the Logical Links between the Sites and the Health Eligibility Center (HEC) are:

Logical Link for transmissions from VAMC to HEC: LLvisn#VISN

Logical Link for transmissions from HEC to VAMC:

At the HEC

HEC specific Logical Link: LLvamc#VAMC

At the Site

The Site can select one of the following two Logical Links to use*:

1. Site Standard Logical Link: VA<site mnemonic>

2. HEC specific Logical Link: LLvamc#VAMC

{where: visn# = VISN (Veterans' Information System Network) number of station.

vamc# = Station number of the receiving/sending VAMC station

site mnemonic = 3 digit code that defines a site Logical Link}

The specific LLvisn#VISN (ex LL2VISN) logical link will be set up at the corresponding VAMC Station and will contain the TCP/IP Address and Port for the HEC (Health Eligibility Center). The LLvamc#VAMC (ex LL516VAMC) will be set up at the HEC for each VAMC Station. The LLvamc#VAMC will contain the TCP/IP Address and Port for the VAMC Station of which vamc# is the Station number.

Examples of logical link set up for station 500:

At the HEC

LL2VISN ...MULTI-LISTENER logical link (station 500 has a VISN=2) with VAMC receiving port number.

LL500VAMC ...CLIENT (SENDER) logical link with IP Address/Port for station 500.

At VAMC Station 500

LL2VISN ...CLIENT (SENDER) logical link with IP Address/Port for the HEC.

*LL500VAMC ...MULTI-LISTENER logical link with VAMC receiving port number.

*VAALN ...MULTI-LISTENER logical link with VAMC receiving port number.

Note: There may be one or more stations which share the same VISN number.

* The Site can choose to use the Site Standard Logical Link or the HEC specific Logical Link.

Site Standard Logical Link

Pros	Cons
<ul style="list-style-type: none"> No new Logical Link to support/maintain For VMS : No new UCX Services or VMS COM files to create For CACHE: No additional jobs running on your system 	<ul style="list-style-type: none"> All HEC transmission traffic will be added to your current queue of HL7 traffic. If the current speed of HL7 transmission processing is not sufficient, the addition of the HEC-specific transmissions to the same queue, will negatively impact this situation.

HEC-Specific Logical Link

Pros	Cons
<ul style="list-style-type: none"> Provides a separate queue for HEC-specific transmissions to be processed. Your current HL7 transmission processing will be minimally affected by the addition of the HEC specific transmissions. 	<ul style="list-style-type: none"> New Logical Link to support/maintain For VMS: New UCX Service and VMS COM file to create For CACHE: Another background process needed to receive transmissions Coordination when going live will be needed to make sure the IP Address and Port are correct.

Routine List with Checksums

The following is a list of routines included in this patch. The second line of each of these routines will look like:

```
<tab>;;5.30;REGISTRATION;**[patch list]**;[date]
```

Routine Name	Before Patch	After Patch	Patch List
DGENEGT2	5421596	5348881	232, 417, 363
DGENQRY1	10286494	10624143	147, 232, 363
DGENUPL	7008110	7160342	147, 222, 232, 363
DGENUPL1	5967784	6082317	147, 222, 232, 314, 397, 379, 407, 363